# 一、快捷操作

## 1.1 视口切换

通过“Alt+G、H、J、K”可观察不同视口下的场景。

## 1.2 场景操作

(1) F ：快速聚焦；

(2) Alt + 拖拽：原样复制；

(3) Shift + 拖拽：沿着轴向移动；

(4) Ctrl + G：打组；

(5) Shift + G：解组；

(6) Ctrl + W：直接复制；

(7) H：隐藏物体；

(8) End：直接让物体附着到地面；

(9) T：更改透明物体的可选择性；

(10) Alt + S：开启模拟；

(11) F8：运行时弹出鼠标。

## 1.3 蓝图中快捷键

(1) 右键拖动；滚轮缩放；

(2) Alt+鼠标左键删除连线；

(3) Ctrl+W：复制节点；

(4) Ctrl+拖拽变量：获取变量（get）；

(5) Alt+拖拽变量：设置变量（set）；

(6) C：添加注释。

# 二、Event节点

[事件（Events） | 虚幻引擎文档 (unrealengine.com)](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/Events/)

事件（event）是从游戏性代码中调用的节点，在事件图表（EventGraph）中开始执行个体网络。它们使蓝图执行一系列操作，对游戏中发生的特定事件（如游戏开始、关卡重置、受到伤害等）进行回应。这些事件可在蓝图中访问，以便实现新功能或覆盖默认功能。

## 2.1 Event Level Reset

Level Reset 事件在关卡重启时发出执行信号。 它在关卡重新加载后进行某项触发时非常实用。 如玩家角色已死亡，但关卡无需重新加载时。

## 2.2 Event Actor Begin Overlap

Actor之间发生了碰撞事件，该事件发生需满足：

(a) Actor 之间的碰撞响应必须允许重叠；

(b) 执行事件的两个 Actor 的 Generate Overlap Events 均设为 true；

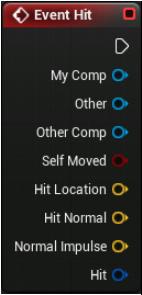
(c) 两个 Actor 开始重叠；两者移到一起或其中一个创建时与另一个重叠。

## 2.3 Event Actor End Overlap

Actor之间的重叠即将停止，需满足与2.2相同的条件。

## 2.4 Event Hit

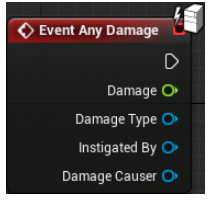
只要其中一个相关 Actor的碰撞设置中 Simulation Generates Hit Events 设为 true，该事件便会执行。

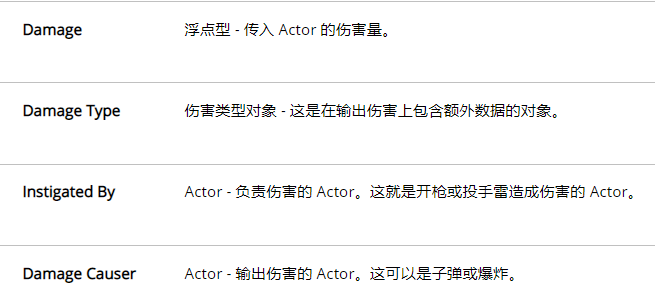




## 2.5 Event Any Damage

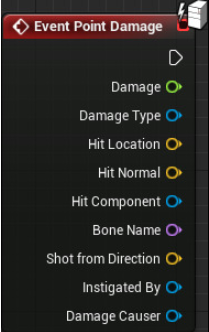
此事件在造成整体伤害时出现。如溺死或环境伤害，非点伤害或放射伤害。





## 2.6 Event Point Damage

点伤害，代表由投射物、扫射武器、甚至近战武器造成的伤害。





## 2.7 Event Radial Damage

放射伤害（Radial Damage）事件在该序列的父 Actor 受到放射伤害时调用。这可用于处理基于爆炸伤害或间接伤害的事件。

## 2.8 Event Actor Begin Cursor Over

使用鼠标界面时，鼠标光标在 Actor 上悬停时执行的事件。

## 2.9 Event Actor End Cursor Over

使用鼠标界面时，鼠标光标在 Actor 上移开时执行的事件。

## 2.10 Event Begin Play

游戏开始时将在所有 Actor 上触发此事件。游戏开始后生成的所有 Actor 上均会立即调用此事件。

## 2.11 Event End Play

Actor 不存在于世界场景中时执行此事件。

## 2.12 Event Destroyed

Actor 被销毁时执行此事件。

## 2.13 Event Tick

游戏进程中每帧调用的简单事件。

# 三、延时

Delay 延时触发，单位为秒。

# 四、碰撞

[碰撞概述 | 虚幻引擎文档 (unrealengine.com)](https://docs.unrealengine.com/4.26/zh-CN/InteractiveExperiences/Physics/Collision/Overview/)

## 4.1 碰撞的几个原则

（1）阻挡 会设置为阻挡的两个（或更多）Actor之间自然发生。但是，需要启用模拟生成命中事件（Simulation Generates Hit Events）才执行事件命中，该功能在蓝图、可破坏物Actor、触发器等处使用，即出发相应事件，进行下步操作。

（2）将Actor设置为重叠往往看起来它们彼此忽略，如果没有生成重叠事件（Generate Overlap Events），则二者基本相同。

（3）对于彼此阻挡的两个或更多模拟对象，它们都需要设置为阻挡相应的对象类型。

（4）对于两个或更多模拟对象：如果一个设置为重叠对象，另一个设置为阻挡对象，则发生重叠，而不会发生阻挡。

（5）即使一个对象会阻挡另一个对象，也可以生成重叠事件，尤其是高速运行的对象。

（6）不建议一个对象同时拥有碰撞和重叠事件。虽然可以，但需要手动处理的部分太多。

（7）如果一个对象设置为忽略，另一个设置为重叠，则不会触发重叠事件。

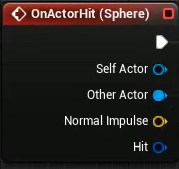
## 4.2 设置

可在“碰撞预设”部分设置Actor的碰撞形式。

## 4.3 碰撞事件

（1） OnActorHit()：触发碰撞事件，括号内为对象（勾选了Simulation Gener

-ates Hit Events的对象）。



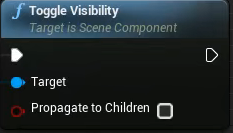
（2）OnActorBeginOverlap()：触发重叠事件。

只有发生重叠的两个Actor都勾选了Generate Overlap Events才触发。

# 五、类蓝图相关

## 5.1Toggle Visibility

修改可见性。



## 5.2 文字组件：TextRender。

## 5.3 Enable Input



接收用户输入。

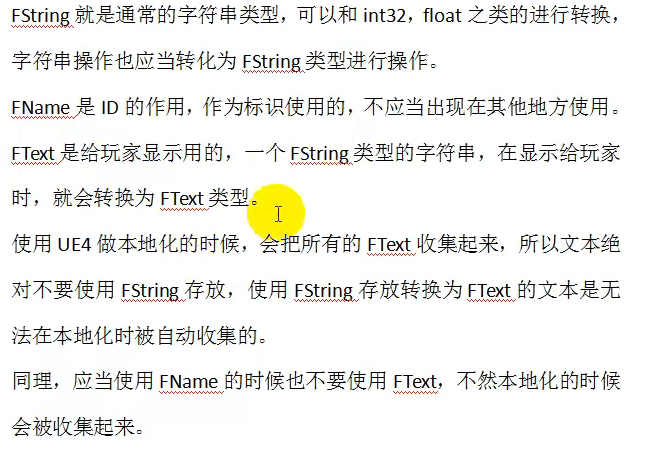
## 5.4 Disable Input

不接收用户输入。

## 5.5 Get Player Controller

获取玩家控制器，即得到当前用户输入的各项信息（鼠标、键盘按键）。

## 5.6 类型差异



## 5.7 Cast To (类型名)

查看输入类型是否能转换为括号中的类型。（起过滤的作用）

## 5.8 Get World Delta Seconds

获取帧速率，即每一帧之间的时间间隔。

## 5.9 Normalize

向量标准化，转换为单位向量。

## 5.10 GetWorldTransform

获取物体的世界坐标。

## 5.11 Break Transform

打散变换，拆分为多个向量。

## 5.12 Make Transform

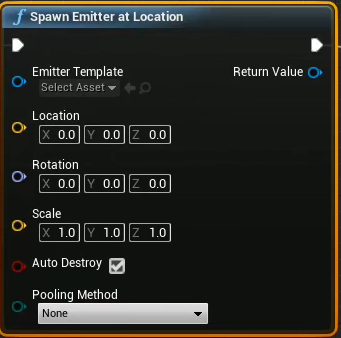
创建变换，输入为多个向量。

## 5.13 SetActorTransform

设置Actor的位置。

## 5.14 Spawn Emitter at Location

在指定位置播放粒子效果。



## 5.15 OnDestoryed()

括号中的类被销毁时发送通知。

## 5.16 Get Actor Forward Vector

获取角色面向。

## 5.17 InputAction (动作名)

虚幻引擎中可在设置中为某些动作预设按键，进而在蓝图中使用“InputAction (动作名)”节点进行触发。

## 5.18 Set Filed Of View

设置相机视角，可用于设计FPS游戏中的开镜操作。通过改变“In Field of View”进行调整。

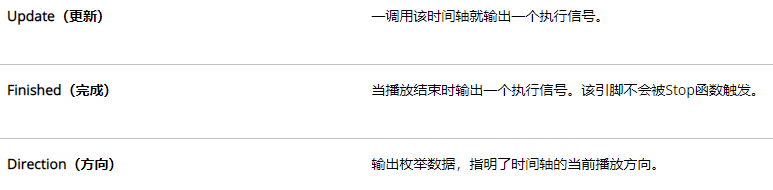


## 5.19 [Timeline](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/Timelines/)

时间线，可用于制作动画（淡入淡出等）；近似看作两种状态的逐渐切换。表现位置的逐渐变换。







## 5.20 Play Sound at Location

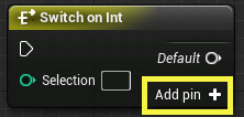
在指定位置播放声音。



# 六、[流程控制](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/FlowControl/)

## 6.1 Switch on Int(string)

类似于C++中的switch语句。

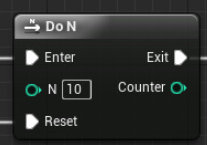


## 6.2 Branch

分支节点，类似于C++中的if…else。

## 6.3 DoN

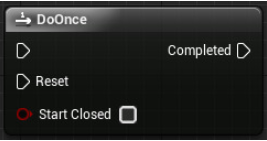
DoN 节点将会 N 次触发执行脉冲。 在达到限制后，它将会停止所有的输出执行，直到脉冲被传入其 Reset （重置）输入。

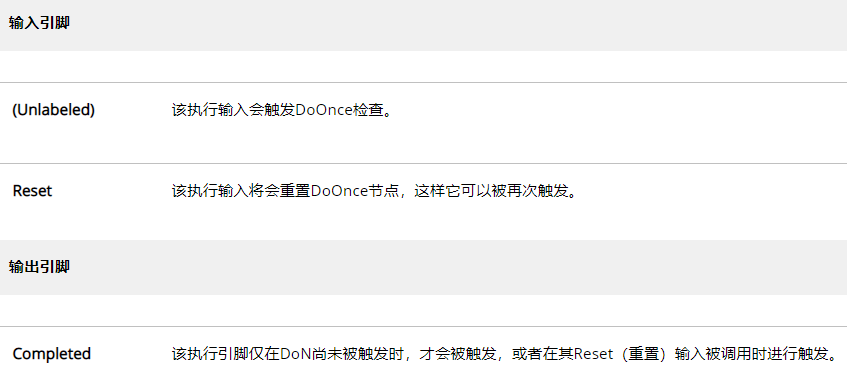




## 6.4 DoOnce

将会仅仅触发执行脉冲一次。 在之后，它将会停止所有的输出执行，直到脉冲被传入其 Reset （重置）输入。 该节点等同于DoN节点中 N = 1 的情况。





## 6.5 FlipFlop

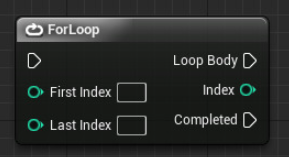
反转节点，节点取入执行输出并在两个执行输出间切换。即它的输出值为：轮流使用输入的值作为输出。





## 6.6 ForLoop

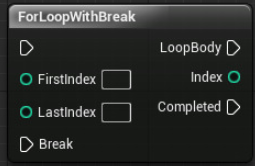
ForLoop节点的工作原理等同于标准的代码循环，将会在开始和结束之间的每个索引触发执行脉冲。（循环迭代将会在不同的帧间发生，这样的话大量循环可能会影响性能表现）





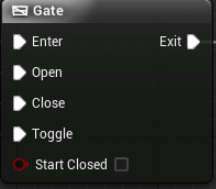
## 6.7 ForLoopWithBreak

ForLoopWithBreak 节点包含了能中断循环的输入引脚，除此之外，它运行的方式与ForLoop节点非常相似。



## 6.8 Gate

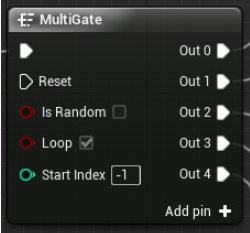
Gate（门） 节点用来开启和关闭执行流。 简单来说，Enter输入取入执行脉冲，同时门的当前状态（开启或关闭）将会决定这些脉冲是否从Exit输出中传出。





## 6.9 MultiGate

MultiGate 节点取入单个数据脉冲并将其传送到任意数量的潜在输出。 这个过程随机按顺序发生，可能会循环。





## 6.10 Sequence（序列）



Sequence （序列） 节点使得单个执行脉冲能按顺序触发一系列事件。 节点可能有任意数量的输出，所有的输出引脚都会在序列节点一获得输入时就被调用。 它们将总是按顺序被调用，但不会有任何延迟。 对一般用户来说，输出引脚看起来好像被同时触发了一样。

## 6.11 WhileLoop

测试条件和主体就是构成 WhileLoop 的全部。在主体中执行语句之前，蓝图计算WhileLoop的测试条件，以确定它是否为true。在主体中执行语句之后，蓝图重新计算测试条件，如果条件仍为true，它将继续在循环主体中执行语句。否则，如果测试条件返回false，则蓝图将终止循环并退出循环的主体。

（类似于C++中的do…while）





# 七、[函数](https://docs.unrealengine.com/4.27/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/Functions/)

## 7.1 概念

函数（Functions）是属于特定蓝图的节点图表，他们可以从蓝图的另一个图表执行或调用（一个蓝图可以有多个图表）。函数有一个有节点指定的单一进入点，函数的名称包含一个执行输出引脚。当从另一个图表调用函数时，输出执行引脚将被激活， 从而使连接的网络执行。

## 7.2 访问说明符

同C++，函数作为蓝图类的成员，可设置其属性为“private”、“protected”、“public”三种。

## 7.3 纯与非纯

函数可以为 纯（Pure） 类型，也可以为 非纯（Impure） 类型。主要区别在于，纯函数承诺不以任何方式修改状态或类的成员，而非纯函数可以自由修改状态。纯函数通常用于只输出数据值的 getter函数或运算符。

（个人理解：C++中，纯类型相当于在函数声明时，函数名称后面加const，仅用于访问类成员）

**7.4 函数功能**

在蓝图图表中点击添加函数后生成对应函数，并开启新选项卡。在“details”部分可对函数的输入、输出进行设计，并通过与蓝图事件列表相似的设计流程在函数中添加节点，实现函数的功能。

## 7.5 调用函数

在完成函数设计的基础上，可在事件列表中对函数进行调用，其使用方式与其他节点的使用方式类似。此外，函数还可在其他外部蓝图类中进行调用，前提是引用了包含要调用函数的蓝图（可通过“cast to 蓝图类”进行调用）

通过此方法也可对蓝图类的public型数据成员进行调用。

# 八、[蓝图接口](https://docs.unrealengine.com/4.27/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/Types/Interface/UsingInterfaces/)

## 8.1 概念

蓝图接口（Blueprint Interface） 是一个或多个函数的集合：只有名称，没有实例可以添加到其他蓝图中。任何添加了该接口的蓝图都保证拥有这些函数。接口的函数可以在添加它的每个蓝图中提供功能。在本质上，这类似于一般编程中的接口概念，它允许多个不同类型的对象通过一个公共接口共享和被访问。简单地说，蓝图接口允许不同的蓝图相互共享和发送数据。

用于定义多个类之间的共性特征，但实现上有所差异。

## 8.2 [调用方法](https://docs.unrealengine.com/4.27/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/Types/Interface/UsingInterfaces/)

在蓝图“类设置”部分对应的“details”中找到“接口”，添加对应的蓝图接口，然后通过类似于函数调用的方式对其进行操作。

在调用接口后，可对其进行修改覆盖。

## 8.3 [使用时机](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/BlueprintCommsUsage/)

数个蓝图中存在一些相似功能，但在调用后执行不同的效果。

（1）玩家拥有一个火焰喷射器，使用时将触发名为 ElementalDamage 的事件。树木蓝图接受 ElementalDamage 调用并将树木烧毁，而雪人蓝图将把雪人融化。

（2）玩家拥有一个"use"按钮，按下按钮可执行开门、关灯、拾起道具等操作。

（3）敌人拥有特殊能力，可基于玩家的体力值完成变化，执行不同操作。

# 九、[蓝图通信](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/BlueprintCommsUsage/)

## 9.1 概念

在蓝图之间传递或共享信息。

## 9.2 [直接蓝图通信](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/BlueprintComms/)

直接蓝图通信（Direct Blueprint Communication） 是最常见的蓝图通信方法，而且当您有两个蓝图并且想要它们在某个时刻彼此交流时，它可以提供良好的效果。这种交流总是一对一的；这意味着一个蓝图（"工作蓝图（Working Blueprint）"）请求访问另一个蓝图（"目标蓝图（Target Blueprint）"）。直接蓝图通信（Direct Blueprint Communication）最简单的使用方法是通过公开的对象[变量（Variable）](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/Variables)获得对目标蓝图的参考，然后指定要访问该蓝图的哪个实例。

具体流程：

（1）在工作蓝图中创建变量，变量类型为目标蓝图的类型，变量设置为可编辑；

（2）在工作蓝图中操作（1）中设置的变量可实现对目标蓝图的操作。

[具体案例](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/BP_HowTo/DirectBlueprintComs/)

# 十、[事件分配器](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/BlueprintCommsUsage/)

## 10.1 概念

事件分配器适用于告知其他“正在倾听的”蓝图已发生事件。事件发生时，正在倾听的蓝图会作出反应，并相互独立的执行预期操作。

（就是某个事件一旦发生，便会触发后续一系列蓝图事件的调用）

## 10.2 [使用时机](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/BlueprintCommsUsage/)

（1）需要从角色蓝图到关卡蓝图进行通信：

玩家角色升级，需要开放之前锁定的区域；

玩家角色按下行动按钮，对关卡执行某种操作；

（2）生成的 Actor 执行某种操作时触发事件：

生成一个 Boss，Boss 被消灭时触发事件，在世界场景中生成一个奖励；

在关卡中生成一个道具（武器等）并在道具被拾起时告知道具和角色。

## 10.3 [创建事件分配器](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/EventDispatcher/)

（1）在“我的蓝图”面板中，单击事件分发器类别上的“+”按钮；

（2）在名称字段中输入事件分发器的名称，该字段显示在“我的蓝图（My Blueprint）”选项卡中列表的末尾。

## 10.4 [具体案例](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/BP_HowTo/EventDispatcher/)（调用方式）

（1）在类蓝图中创建事件分配器；

（2）将事件分配器对应的Call节点与要引发事件分配器的事件相连；

（3）在关卡蓝图中创建对象的引用，将其转换为创建事件分配器的类，并创建“assign 事件分配器名称”节点；

（4）将要触发的事件与对应事件节点相连，实现事件分配。

# 十一、[蓝图投射](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/BlueprintCommsUsage/)

## 11.1 概念

蓝图投射的形式为使用Cast节点，使用Cast节点便是在尝试检查发出转换的对象是否为被转换的特定对象；若是，则进行访问；反之，则忽略请求。

## 11.2 使用时机

（1）需要访问另一个蓝图的特殊版本：

角色走进火焰中，导致体力值耗尽。

投射到特殊的角色蓝图，以便访问并变更体力值。

角色死亡，需要重新生成。

投射到特殊的游戏模式蓝图，执行重新生成脚本。

（2）需要访问相同类的多个蓝图，并以相同方法进行修改：

场景中拥有数盏灯，事件发生时需将它们开启或关闭。

投射到灯蓝图并执行函数将灯关闭。

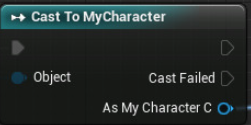
（3）需要访问一个特殊的子蓝图：

存在基于一个动物蓝图（猫、狗、鸟）的数个蓝图，需要访问其中一个动物。

投射到猫、狗和鸟，访问其相应的蓝图和特有功能。

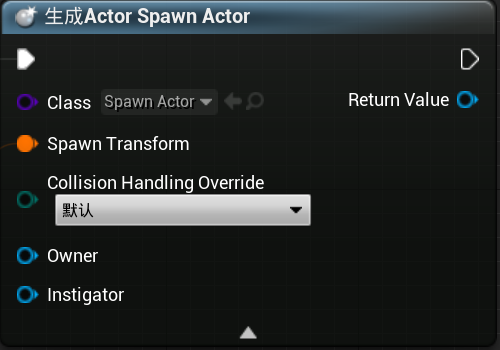
## 11.3 [使用方式](https://docs.unrealengine.com/4.26/zh-CN/ProgrammingAndScripting/Blueprints/UserGuide/CastNodes/)

可通过Cast to 其他类型中的“As …”引脚访问相应类中的成员。



# 十二、[生成/销毁Actor](https://docs.unrealengine.com/4.27/zh-CN/ProgrammingAndScripting/SpawnAndDestroyActors/)

关键节点：



跳过在Class中选择要生成的类型来实现。

# 十三、定时器



存在多种定时器，有触发事件的、触发函数的，区别在于左侧引脚。“Time”表示每隔多久执行一次“Event”，“Looping”代表是否循环。